Supporting Knowledge Collaboration Using Social Networks in a Large-Scale Online Community of Software Development Projects

Masao Ohira Tetsuya Ohoka Takeshi Kakimoto Naoki Ohsugi Ken-ichi Matsumoto
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara, JAPAN 630-0192
{masao, tetsuy-o, takesi-k, naoki-o, matumoto}@is.naist.jp

Abstract

The scale-free network shown in the small world phenomenon indicates that our human society consists of a small number of people who play the role of hubs linked with many nodes (persons) and a large number of people as nodes linked with few nodes. From our analysis of a large-scale online community—SourceForge.net which has a large number of developers and projects, we have found that SourceForge also exists as a scale-free network. That is, only a minority of developers join many projects and have rich links with other developers, while the majority join few projects and have very limited social relations with others. The goal of our study is to build a system that supports knowledge collaboration in a large-scale online community of software development projects. In this paper, we discuss the challenges of supporting knowledge collaboration in such a large online community that is a scale-free network and then introduce the prototype system called D¬SNS (Dynamic Social Networking System).

1. Introduction

In general, technologies used in software development have been evolving from day to day. Although software developers continuously require a considerable amount of new and diverse knowledge about relevant technologies, it is not realistic to try to support developers so that they become knowledgeable about all the relevant technologies. It would be important to provide a mechanism that enables developers to mutually share and create useful knowledge for software development, available where and when needed.

Successful open source software (OSS) communities are a good example of knowledge collaboration. Developers in an OSS community discuss technological problems and reported bugs; share useful information and new ideas; and create OSS products. In a large-scale online community of software development projects such as SourceForge.net and Java.net, however, a number of developers are confronted with the difficulty of asking for help from others [8].

For instance, SourceForge has a large number of developers and projects but the majority of the projects only have few developers (less than 3 developers). Due to the lack of supporting cross-project knowledge collaboration, the small projects with few developers often cannot gather sufficient contributors such as bug reporters and new developers, and then remain sleeping or dead in some cases. As contrasted with the small projects, the large projects with many contributors are easier to gather further contributors (known as the "rich-get-richer" effect) [6].

The recent studies on social networks revealed that our human society consists of a small number of people who play the role of hubs linked with many nodes (persons) and a large number of people as nodes linked with few nodes—the scale-free network[2, 12]. The results of the analysis of SourceForge [6, 8] indicate that SourceForge also exists as a scale-free network. The issue mentioned above can be regarded as one of characteristics of the scale-free network.

The goal of our study is to build a system that supports knowledge collaboration in a large-scale online community of software development projects. There are two issues that must be resolved in designing a support system: the lack of mechanisms for supporting cross-project knowledge collaboration and the negative effects in a community with a scale-free network. In the next section, we discuss the challenges of supporting knowledge collaboration in such the community with a scale-free network structure. Section 3 describes our approach to deal with the issues and then introduce the prototype system called D-SNS

(Dynamic Social Networking System) in Section 4. Section 5 discusses the advantage and the disadvantage of our approach, comparing to an existing study of knowledge collaboration.

2. Challenges of supporting knowledge collaboration

In this section, we discuss the challenges of supporting knowledge collaboration in a large-scale community of software development projects, especially from the perspectives of the organizational structure and the network structure of social relationships in the community.

2.1. Organizational structure

Large-scale online communities such as SourceForge.net and Java.net provide OSS projects with free hosting and a variety of development support tools (e.g., repositories for source code, bug tracking system, tools for open discussions, etc.), which can be seen as knowledge repositories for software development.

Figure 1 shows the typical organizational structure in a large-scale online OSS community. Each project can accumulate, share, and exploit own past knowledge for creating own software products, using tools provided by the community. However, because the community dose not provide each project with tools or mechanisms for sharing the aggregate knowledge in the community, it is difficult for developers and other contributors to find and use it, even if a good deal of useful knowledge exists in the community. For the same reason, developers (D_A) are confronted with the difficulty of asking for help from developers of other projects (D_B) and contributors (C), even if D_B and C may possess knowledge which D_A would like to know but none of D_A have.

These issues caused by the typical organizational structure in a large-scale online OSS community could be critical, because a considerable proportion of projects in the community (e.g., over 80% of all projects in SourceForge) often consist of less than 3 software developers [8]. Support mechanisms for cross project knowledge sharing and knowledge collaboration should be provided especially to a number of small projects with few developers which tend to be the lack of knowledge resources for creating software products.

2.2. Network structure of social relationships

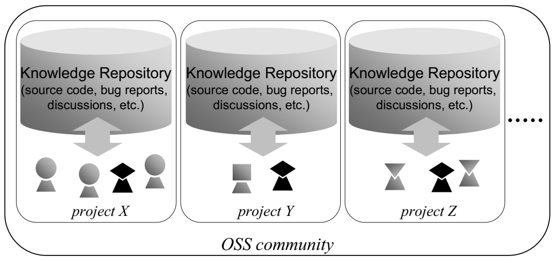

Knowledge Repository
(source code, bug reports, discussions, etc.)
Knowledge Repository
(source code, bug reports, discussions, etc.)
Knowledge Repository
(source code, bug reports, discussions, etc.)
.....
project X
project Y
project Z
OSS community

Figure 1. Typical organizational structure of a large-scale online OSS community

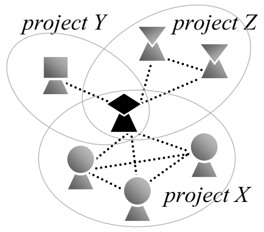


Figure 2. Network structure of social relationships in a community

Using social relationships among developers in an OSS community would be a possible solution to the issues described in Section 2.1. For instance, the blacked-out developer in Figure 2 is participating in the three projects (X, Y, and Z). All the developers of the three projects can be seen to be socially connected via her/him within 2 degrees of separation as Figure 2. Because the blacked-out developer who connects all the developers as one cluster is likely expected to "know what kind of knowledge exists in which project" and "who is knowledgeable about what," the other developers are able to effectively acquire knowledge necessary for creating software products by inquiring of the blacked-out developer.

However, if only few developers in a community are participating in multiple projects as illustrated in Figure 1, they would bear a heavy burden of many inquiries from other developers. The recent studies on OSS communities reported that OSS development largely depends on contributions not by burden-sharing among developers but by a small portion of developers. For instance, only 4% of developers in the Apache community created 88% of new code and fixed 66% of defects [7]. From a total of 196 developers in the Ximian project, 5 developers account for 47% of the modification requests (MRs) [4]. 4% of members account for 50 %s of answers on a user-to-user help site [5].

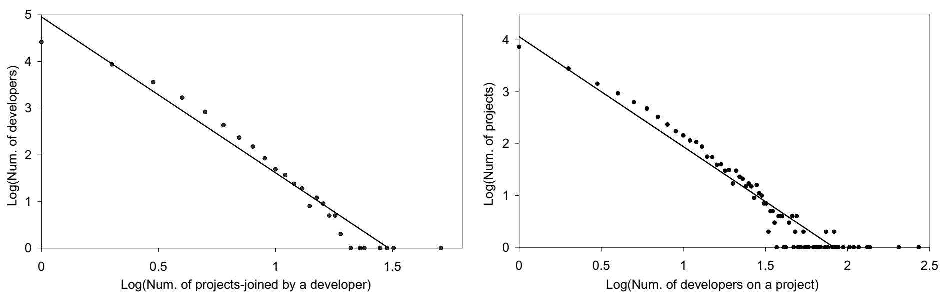


Figure 3. Power low relationships of the scale-free network in SourceForge.net (Feb. 2005)

From our analysis [1] of social networks in SourceForge, we have found that SourceForge exists as a scale-free network [2]. In a scale-free network, what makes the world so small is the existence of hubs [12]. Social relationships in a scale-free network can be graphed as power low relationships as Figure 3. The left graph in Figure 3 shows how many developers (Y axis) participated in how many projects (X axis). The right shows how many projects (Y axis) have how many developers in their projects (X axis).

Roughly speaking, these results also imply that only a minority of developers as hubs connects developers with developers in SourceForge and have the possibility of tying developers and projects together as the aggregate knowledge repository. Although we described the need of supporting cross project knowledge collaboration for a number of small projects with few developers in Section 2.1, the issues caused by the network structure of social relationships in a large-scale community also indicate that the support mechanism that prevents a pile of inquiries rushing to the "hub" developers is needed.

3. Approach

This section describes our approach to construct mechanisms which support cross project knowledge collaboration for a number of projects with few developers and protect a small number of "hub" developers against a large amount of requests from other developers in a large-scale online community.

3.1. Constructing social networks

As we mentioned in Section 2.2, exploiting social networks among developers is a possible solution for supporting cross-project knowledge collaboration in a community with a lot of independent projects.

Constructing social networks: Using the method of social network analysis [10, 11], we can define social relationships among developers in a large-scale online community. The relationships as illustrated in the social network in Figure 2—"who belongs to which projects" can be represented as affiliation networks [11]. 25% of all the developers in SourceForge, for instance, can be connected as one cluster (knowledge network) by defining social relationships among developers as affiliation networks. Using the notion of affiliation networks, we can construct multiple clusters that can be seen as multiple knowledge networks in a community.

Unifying clusters and independent nodes: Furthermore, we can construct the unified knowledge network consisting of multiple clusters and independent nodes (projects with one developer) in a community. If a developer belonged to a cluster (C_A) or an independent node (N) has used a repository of another cluster (C_B), we can also define that C_A , C_B , and N are connected. By doing so, most of developers belonged to any projects can be unified into one large cluster. Active users who do not belong to any projects but have input their knowledge into a project repository (e.g., bug reports) are also able to include in the cluster.

3.2. Protecting "hub" developers

The key of constructing the large knowledge network (cluster) heavily depends on the existence of a minority of "hub" developers in a large community. It is natural that the majority of developers with few links is attracted to "hub" developers who are likely knowledgeable about a variety of technologies and other developers. However, "hub" developers would bear a heavy burden of many inquiries from "node" developers, if there are some means of protecting "hubs." If not so, "hubs" may leave the community [3].

[1] The analysis of SourceForge.net was conducted in Feb. 2005. We analyzed the data of over 90,000 projects and 130,000 developers participating in more than one project.

Because our approach to supporting cross-project knowledge collaboration depends on "hubs," we have to avoid such the situation absolutely.

Ye et al. [13] proposed a unique communication mechanism to resolve the issue above. All the members in a community allow sending messages (requests) to the community but they do not know who will receive the messages. The members relevant to senders and senders' tasks will receive the messages. Because a sender of a message cannot know who is a receiver, a receiver of the message dose not need to feel mental pressure from replying the message.

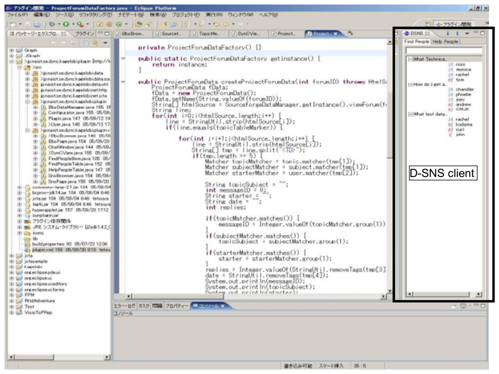
In addition to this mechanism, we propose a mechanism to cultivate social relationships among developers with few or no links. Because developers with only few links are less likely to have help from others sufficiently, it is necessary to provide them with some means so that they increase social connections to others. In the next section, we describe the mechanism for constructing social relationships effectively.

4. D-SNS: Dynamic Social Networking System

The section describes the overview of D-SNS (Dynamic Social Networking System), the mechanism of cultivating social relationships among developers, and an example of the use of D-SNS.

4.1. Overview of D-SNS

The right list in Figure 4 shows a user interface of a D-SNS client. Currently, we have been implementing the system as an Eclipse plug-in in order to support developers of SourceForge. Figure 5 illustrates the architecture of D-SNS. D-SNS collects affiliation information of each developer from SourceForge and uses it for creating and managing social relationships among developers. D-SNS also acquires information on knowledgeable developers (e.g., who is knowledgeable about what, etc.) by extracting technical terms and important words from communication logs, user registration info., bug reports, and so forth that are stored in projects' repositories of SourceForge. D-SNS constructs knowledge networks using the information of affiliation networks and identifies knowledgeable developers. Here, "knowledgeable" dose not mean the absolute amount of knowledge of developers but is defined by the relative amount of knowledge among developers [13]. That is, a developer (D_A) is knowledgeable about something for a developer (D_B) at a certain point, but D_B can be knowledgeable about something for D_A at another point.


```
private ProjectForumDataFactory() {}

public static ProjectForumDataFactory getInstance() {
    return instance;
}

public ProjectForumData createProjectForumData(int forumID) throws HtmlSc
    ProjectForumData fData;
    fData = new ProjectForumData();
    fData.setName(String.valueOf(forumID));
    String[] htmlSource = SourceforgeDataManager.getInstance().viewForum(fo
    String line;
    for(int i=0;i<htmlSource.length;i++) {
        line = StringUtil.strip(htmlSource[i]);
        if(line.equals(topicTableMarker)) {

            for(int j=i+1;j<htmlSource.length;j++) {
                line = StringUtil.strip(htmlSource[j]);
                String[] tmp = line.split("<TD>");
                if(tmp.length == 5) {
                    Matcher topicMatcher = topic.matcher(tmp[1]);
                    Matcher subjectMatcher = subject.matcher(tmp[1]);
                    Matcher starterMatcher = user.matcher(tmp[2]);

                    String topicSubject = "";
                    int messageID = 0;
                    String starter = "";
                    String date = "";
                    int replies;

                    if(topicMatcher.matches()) {
                        messageID = Integer.valueOf(topicMatcher.group(1))
                    }
                    if(subjectMatcher.matches()) {
                        topicSubject = subjectMatcher.group(1);
                    }
                    if(starterMatcher.matches()) {
                        starter = starterMatcher.group(1);
                    }
                    replies = Integer.valueOf(StringUtil.removeTags(tmp[3]
                    date = StringUtil.removeTags(tmp[4]);
                    System.out.println(messageID);
                    System.out.println(topicSubject);
```

Figure 4. D-SNS client on Eclipse

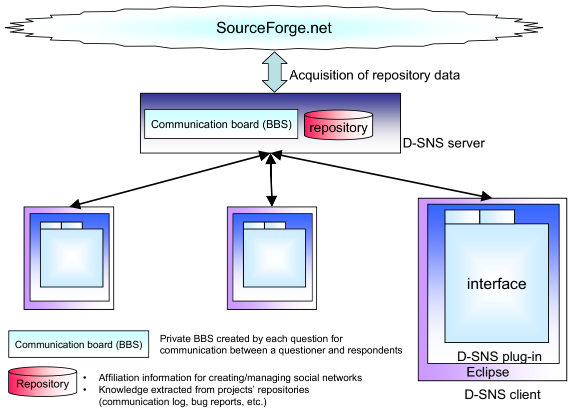

SourceForge.net
Acquisition of repository data
Communication board (BBS)
repository
D-SNS server
interface
D-SNS plug-in
Eclipse
D-SNS client
Communication board (BBS)
Private BBS created by each question for communication between a questioner and respondents
Repository
Affiliation information for creating/managing social networks
Knowledge extracted from projects' repositories (communication log, bug reports, etc.)

Figure 5. D-SNS architecture

This mechanism enables a user of D-SNS to search developers2 who possess relevant knowledge if a user have participated in some projects. Even if a user has participated in one project with one developer (herself) or has not participated in any projects, she can use the system and ask a question. If someone answers it, a social relationship between the respondent and her is established in the system. She will be able to use the social network for searching more developers in the next.

4.2. Cultivating social networks

D-SNS helps a user chose whom he should communicate with, according to a state of a user's social network and a user's role. The system's recommendations are determined by calculating results of keyword matching, the density of a user's social

Table 1. Basic rules for recommending users to appropriate developers

State	User role	Direction	Recommending order	Main effect
R1	Respondent	open→closed	S2→S1→U	Using own skills and knowledge for others
R2	Respondent	closed→open	U→S1→S2	Building up a new network of connections
Q1	Questioner	open→closed	S2→S1→U	Acquiring the higher quality of knowledge
Q2	Questioner	closed→open	U→S1→S2	Acquiring a good deal of knowledge

open: open network, closed: closed network
open→closed: the transition from an open network to a closed network
closed→open: the transition from a closed network to an open network
S1: developers within 1 degree of separation from own or developers directly communicated with before
S2: developers within 2 degrees from own (1 degree of separation from S1)
U: not developers but active users in SourceForge.net, or developers with more than 3 degrees of separation from own

network, and the frequency of communications with others.

Keyword matching: The system evaluates results of keyword matching between technical words stored in the system and users' input keywords. If results of other evaluations are same, the system recommends users to developers with high scores.

Density of a users' social network: The system evaluates a state of a users' social networks based on the density of his network. In general, a closed network is a small, closed, highly similar, and introverted network with higher density [1]. On the contrary, an open network is a large, opened, various, and extroverted network with lower density. The system selects one of the rules in Table 1 based on the density of a user's social network, and then recommends S1, S2, and U to users in the order of the rules. The basic idea is that the system recommends users to appropriate developers so that users with high density (closed network) have an open network and users with low density (open network) have a closed network.

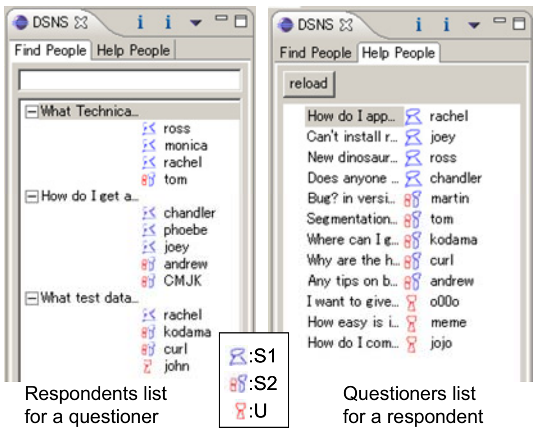
Frequency of communications: The system evaluates the frequency of communications among users because a respondent who communicates with a questioner seems to have better understandings of questioner's demands. Because S2 or U became S1 when a user communicates with them, the system often recommends S1 to a user preferentially.

Finally, a user selects some of developers recommended by D-SNS and communicates to them using BBS.

4.3. An example of using D-SNS

A questioner can ask a question from the "Find People" tab in the left of Figure 6. If someone replies the question, a list of respondents will appear as the list in the left of Figure 6. In a similar way, a respondent can find questioners who would like to know her/his knowledge or acquaintances from the "Help People" tab in the right of Figure 6. If s/he wants to tell a questioner something s/he knows, she can reply to the questioner using BBS (Figure 7). If she does not reply any questions, none of questioners can know that.

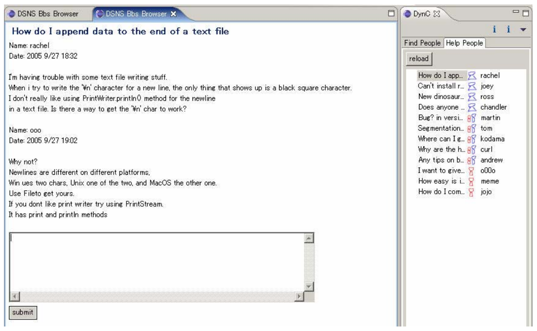
A questioner who received replies selects respondents and communicates to them using BBS in Figure 7. If the questioner is satisfied with the answers from respondents, the communication log including the answers is stored in respondents' repositories as new knowledge and communication history.


DSNS
Find People
Help People
reload
Respondents list for a questioner
Questioners list for a respondent
:S1
:S2
:U

What Technica...
ross
monica
rachel
tom
How do I get a...
chandler
phoebe
joey
andrew
CMJK
What test data...
rachel
kodama
curl
john

How do I app...	rachel
Can't install r...	joey
New dinosaur...	ross
Does anyone ...	chandler
Bug? in versi...	martin
Segmentation...	tom
Where can I g...	kodama
Why are the h...	curl
Any tips on b...	andrew
I want to give...	o00o
How easy is i...	meme
How do I com...	jojo

Figure 6. recommended user lists in DSNS



How do I append data to the end of a text file
Name: rachel
Date: 2005 9/27 18:32

I'm having trouble with some text file writing stuff.
When i try to write the '¥n' character for a new line, the only thing that shows up is a black square character.
I don't really like using PrintWriter.println() method for the newline
in a text file. Is there a way to get the '¥n' char to work?

Name: ooo
Date: 2005 9/27 19:02

Why not?
Newlines are different on different platforms,
Win ues two chars, Unix one of the two, and MacOS the other one.
Use Fileto get yours.
If you dont like print writer try using PrintStream.
It has print and println methods

How do I app...	rachel
Can't install r...	joey
New dinosaur...	ross
Does anyone ...	chandler
Bug? in versi...	martin
Segmentation...	tom
Where can I g...	kodama
Why are the h...	curl
Any tips on b...	andrew
I want to give...	o00o
How easy is i...	meme
How do I com...	jojo

Figure 7. Bulletin board in DSNS

5. Discussions

ReachOut [9] is a chat-based system for peer support, collaboration, and community building. It has a communication mechanism similar to [13] and us, in which users cannot ask a question to a specific person and be guaranteed to receive an answer but can send it to a community or group. ReachOut dose not use knowledge repositories but requires users to make a profile from a checklist such as a category of knowledge before using the system. This approach might not work well for software developers and would require them to update own profiles frequently, since about what users are knowledgeable changes from day to day especially in software development. One of the advantages of using D-SNS is that profiles are not created by users themselves but created by the system automatically, using knowledge accumulated into repositories through users' daily software development.

However, we need to reconsider and improve the mechanisms for identifying knowledgeable developers. Is it sufficient or appropriate to identify them only by using knowledge resources stored in SourceForge? Are there any resources and methods useful for finding more knowledgeable developers? We would like to evaluate the mechanisms in the future. We also need to evaluate the mechanisms for cultivating and managing users' social networks. Are there any essential differences of effectiveness between D-SNS and existing systems (e.g., ML) that allows users to send requests to all members in a community? Generally speaking, members in OSS communities have the spirit of helping one another culturally. OSS developers might prefer connecting anyone to protecting someone ("hub" developers).

6. Future Work

In addition to the evaluations of the system described in the last section, we will implement a browser as an Eclipse plug-in. Browsing a web page of SourceForge on Eclipse make seamless development possible. Furthermore, we would like to implement a visualization capability of social networks [8] to browser so that users understand own social relationship objectively. It would be helpful for determing that they need more connections to other or not. Finally, we have a plan to release D-SNS into the public for evaluations of system's effectiveness.

Acknowledgement: This work is supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan, Grant-in-aid for 15103 of the Open Competition for the Development of Innovative Technology program and for Young Scientists (B), 17700111, 2005.

References

[1] W. E. Baker. *Achieving Success Through Social Capital*. John Wiley & Sons Inc., 2000.

[2] A. Barabasi. *Linked: How Everything Is Connected to Everything Else and What It Means*. Penguin, 2003.

[3] G. Beenen, K. Ling, X. Wang, K. Chang, D. Frankowski, P. Resnick, and R. E. Kraut. Using social psychology to motivate contributions to online communities. In *Proceedings of the conference on Computer supported cooperative work (CSCW'04)*, pages 212–221, Chicago, Illinois, USA, 2004. ACM.

[4] D. German and A. Mockus. Automating the measurement of open source projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, pages 63–67, Portland, Oregon, 2003.

[5] K. R. Lakhani and E. von Hippel. How open source software works: "free" usertouser assistance. *Research Policy*, 32(6):923–943, June 2003.

[6] G. Madey, V. Freeh, and R. Tynan. The open source software development phenomenon: An analysis based on social network theory. In *Americas Conference on Information Systems (AMCIS2002)*, pages 1806–1813, Dallas, TX, 2002.

[7] A. Mockus, R. Fielding, and J. D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(3):309–346, 2002.

[8] M. Ohira, N. Ohsugi, T. Ohoka, and K. Matsumoto. Accelerating cross-project knowledge collaboration using collaborative filtering and social networks. In *Proceedings of 2nd International Workshop on Mining Software Repositories (MSR2005)*, pages 111–115, St. Louis, MO, 2005.

[9] A. Ribak, M. Jacovi, and V. Soroka. "ask before you search": peer support and community building with reachout. In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'02)*, pages 126–135, New York, NY, 2002. ACM.

[10] J. Scott. *Social Network Analysis: A Handbook*. SAGE Publications, 2000.

[11] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[12] D. J. Watts. *Six Degrees: The Science of a Connected Age*. W W Norton & Co Inc, 2003.

[13] Y. Ye, Y. Yamamoto, and K. Kishida. Dynamic community: A new conceptual framework for supporting knowledge collaboration in software development. In *Proceedings of 11th Asia Pacific Software Engineering Conference (APSEC'04)*, pages 472–481. IEEE, 2004.